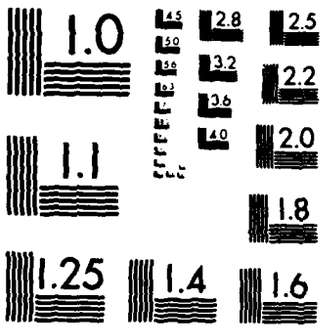
AD-A170 182 ASYNCHRONOUS FINITE STATE MACHINES SIMULATIONS WITH IMPOSED PROCESSING CO (U) NORTHEASTERN UNIV BOSTON MASS DEPT OF ELECTRICAL ENGINEERING 1/1

UNCLASSIFIED S Y KWONKAM ET AL 01 APR 86 F/G 9/2 NL



8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
AFOSR	NM	F49620-82-C-0080

8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
Bolling AFB DC 20332-6448	61102F	2304	A1	

11. TITLE (Include Security Classification)

'ASYNCHRONOUS FINITE STATE MACHINES: SIMULATIONS WITH IMPOSED PROCESSING CONSTRAINTS

12. PERSONAL AUTHOR(S)

M.E. Kolinski and A.T. Miller S.Y. Kwankam, T. L. Johnson

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Reprint	FROM _____ TO _____	1 April 1986	2

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The action of computer control systems on continuous-time discrete-state processes can be accurately represented by asynchronous finite-state machines, and, in particular, a subclass of these machines termed "simple asynchronous machines", or SAMs. To understand the role that practical signal processing constraints may play in characterizing SAM behavior, a simulator capable of incorporating such constraints has been written. The architecture of this simulator and examples of its use are presented.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
DR. MARC JACOBS	(202) 767-4940	NM

ASYNCHRONOUS FINITE STATE MACHINES: SIMULATIONS WITH IMPOSED PROCESSING CONSTRAINTS *

S. Y. Kwankam
ENSP
Universite de Yaounde
Yaounde, Cameroun

M. E. Kaliski and A. T. Miller
Department of Electrical Engineering
Northeastern University
Boston, Massachusetts

T. L. Johnson
General Electric Company
Schenectady, New York

ABSTRACT

The action of computer control systems on continuous-time discrete-state processes can be accurately represented by asynchronous finite-state machines, and, in particular, a subclass of these machines termed "simple asynchronous machines", or SAMs. To understand the role that practical signal processing constraints may play in characterizing SAM behavior, a simulator capable of incorporating such constraints has been written. The architecture of this simulator and examples of its use are presented.

SIMPLE ASYNCHRONOUS MACHINES (SAMs)

The SAM represents a departure from traditional representations of asynchronous finite automata [1], which are typically circuit-based [2], [3], [4] and is more in concert with event-driven models of finite-state machines [5]. It has a finite number of inputs and outputs and a finite number of states. It consists (Figure 1) of three main components: a set of digital function generators (DFGs), which generate asynchronous patterns of state values with respect to time; a multiplexer (MUX), which for a given set of input levels and a given state value, selects one of these patterns, and a readout map, called a logic function (LF), which assigns output values for every combination of input level and state value. The input to the MUX's select lines is triggered by changes in the input levels; the state value thus strobed in is the most recent state value prior to the input changes. Continuity of state is not assumed, but is available as an optional restriction of the simulator.

When a SAM is in a given state and is activated by a "start" signal, the specific input values present trigger an asynchronous pattern of state values and output values with respect to time. As inputs change, various asynchronous patterns of state values and output values occur.

* This work is supported, in part, by the United States Air Force Office of Scientific Research, AFSC, Contract F49620-82C-0080.

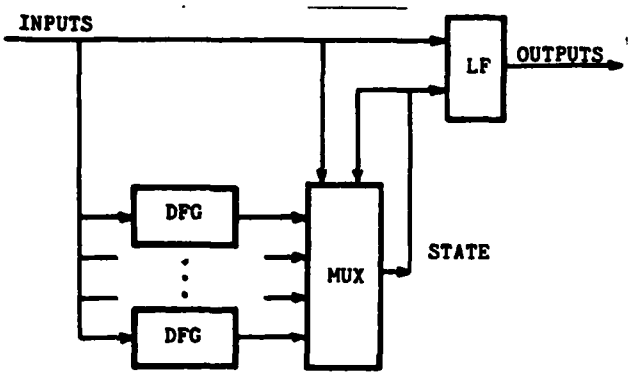


FIGURE 1. THE SAM ARCHITECTURE

SIMULATOR ARCHITECTURE

The simulator stores all asynchronous patterns as hybrid stacks, i.e. sets of two arrays, one with integer entries (representing input, state and output levels), the other with non-negative real values corresponding to the "start times" of these integer levels.

The simulator admits two options. The first, continuity of state, is obtained by requiring that the first state level of the DFG triggered in is equal to the state strobed into the MUX's select lines. Physical processing constraints can be modeled in the simulator as well. These constraints model the time it takes for a system to recover from input changes, and is simulated using a "masking" interval of length T. Any input change occurring within time T of another is ignored for the duration of the masking interval. Thus if the change persists for at least time T it will be recognized; otherwise it will be ignored. This option is extremely valuable as a means of representing synchronization errors in input functions.

The simulator is completely event-driven; hence when there are no further changes in the input lines, the state value and output value become constant as soon as the asynchronous pattern in the last-triggered-in DFG does.

DISCUSSION

To demonstate the capabilities of the SAM simulator, a 74LS93 4 bit binary ripple counter is modeled. To keep the illustrations as simple as possible, only 3 of the 4 bits are used.

Figure 2 is a typical DFG waveform. (The time scale is much finer than for subsequent plots).

Figure 3 is the input to the simulated ripple counter. Note that the input is an asynchronous signal with pulses of various widths. The effect of a non-zero (T=0.5) masking interval on the input seen by the simulator is also shown.

Figure 4 is a plot of the state of the simulated ripple counter with a masking interval, T=0. Inputs are immediately recognized and processed by the SAM simulator. Note that state changes are triggered only by the falling edge of the input waveform. Also note that after the falling edge of the input, it takes some time for the state to reach a stable value. This accurately models the ripple counter behavior.

The output of the simulator is not shown, but for this simple example, is assumed to be identical to the state function.

In Figure 5 is the state function generated by the simulator for a masking interval T=0.5. Observe that the state function is much different from Figure 4.

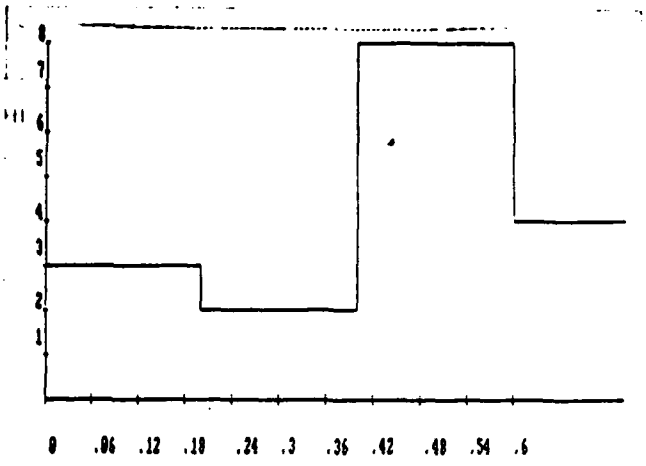


FIGURE 2. AN EXAMPLE DFG

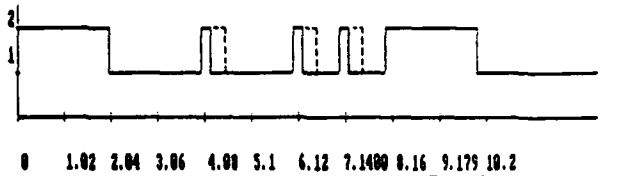


FIGURE 3. INPUT FUNCTION

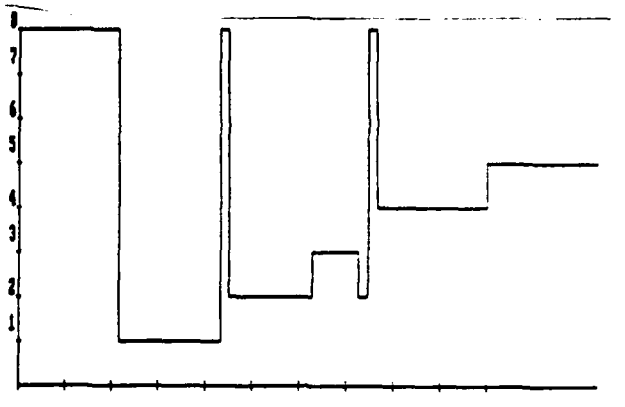


FIGURE 4. STATE FUNCTION, T = 0

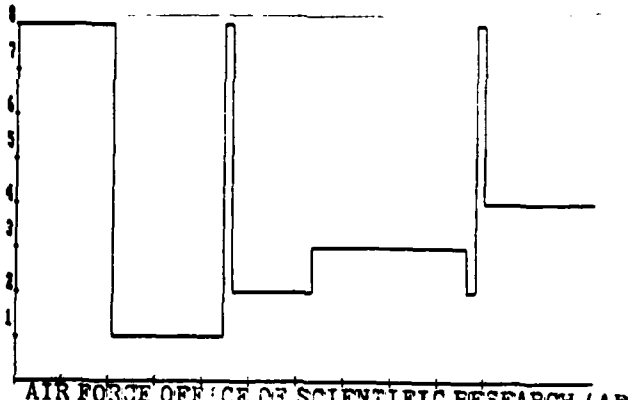


FIGURE 5. STATE FUNCTION, T = 0.5

CONCLUSIONS

The SAM architecture presented is capable of simulating a variety of asynchronous finite-state machines (a UART has also been modeled). The ripple counter example demonstrates how the simulator can be used to explore the effects of various physical processing constraints on the behavior of a SAM.

REFERENCES

(1) Johnson, T. L. and Kaliski, M. E. "Realization of Asynchronous Finite-State Machines", Proc. 22nd IEEE Conf. on Decision and Control, San Antonio, December 1983.

(2) Miller, R. E., Switching Theory (Vol. II), J. Wiley & Sons, Inc., New York, 1966.

(3) Unger, S. H., Asynchronous Sequential Switching Circuits, J. Wiley & Sons, Inc., New York, 1969.

(4) Kohavi, Z., Switching and Finite Automata Theory, McGraw-Hill Book Company, New York, 1978.

(5) Peterson, J. L., "Petri Nets", ACM Computing Surveys, Vol. 9, No. 3, pp. 223-252, 1977.

END

DTIC

9—86